

06-01-00

EK48354847505  
5/31/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Docket No. AUS000153US1

Assistant Commissioner for Patents  
Washington, D.C. 20231

Sir:

Transmitted herewith for filing is the patent application of Inventor(s):  
**Kurt Russell Taylor**

For: **A METHOD AND APPARATUS FOR BRIDGING SERVICE FOR STANDARD  
OBJECT IDENTIFIER BASED PROTOCOLS**

Enclosed are also:

- ☒ 21 Pages of Specification including an Abstract
- ☒ 12 Pages of Claims
- ☒ 6 Sheet(s) of Drawings
- ☒ A Declaration and Power of Attorney
- ☒ Form PTO 1595 and assignment of the invention to IBM Corporation

**CLAIMS AS FILED**

FOR	Number Filed		Number Extra		Rate		Basic Fee (\$690)
Total Claims	57	-20 =	37	X	\$ 18	=	\$666
Independent Claims	6	-3 =	3	X	\$ 78	=	\$234
Multiple Dependent Claims	0			X	\$260	=	\$0
<b>Total Filing Fee</b>							<b>= \$1,590</b>

- ☒ Please charge \$1,590 to IBM Corporation, Deposit Account No. 09-0447.
- ☒ The Commissioner is hereby authorized to charge payment of the following fees associated with the communication or credit any over payment to IBM Corporation, Deposit Account No. 09-0447. A duplicate copy of this sheet is enclosed.
  - ☒ Any additional filing fees required under 37CFR § 1.16.
  - ☒ Any patent application processing fees under 37CFR § 1.17.

Respectfully,

Volel Emile

Reg. No. 39,969  
Intellectual Property Law Dept.  
IBM Corporation  
11400 Burnet Road 4054  
Austin, Texas 75758  
Telephone: (512) 823-1005

15704 U.S. PTO  
05/31/00

15520 U.S. PTO  
09/583411  
05/31/00

05583411-053100

## BACKGROUND OF THE INVENTION

## 1. Technical Field

10

## 15

20

25

30

Docket No. AUS000153US1

component. SNMP is a simple protocol as it does not contain relations like the two protocols discussed next.

Lightweight Directory Access Protocol (LDAP) is another example of a management protocol that uses SMI  
5   OIDs. It is a simplified version of the X.500 standard. LDAP support is implemented in Web browsers and e-mail programs, which can query an LDAP-compliant directory. Queries in LDAP can be complex, such as the query for what printer objects can be accessed by a particular user  
10   object.

The Common Information Model (CIM) also describes management information in an OID format. CIM is implementation independent, allowing different management applications to collect the required data from a variety  
15   of sources. CIM includes schemas for systems, networks, applications and devices, among others. It also provides data mapping definitions for the use of SNMP data from within a CIM schema.

Currently if a server is processing queries using a  
20   variety of protocols, it must keep a separate repository of objects for each protocol. Often these repositories contain references to the same object, albeit with a different object identifier. If an object is being changed in different ways through two or more protocols,  
25   this can lead to inconsistencies. A single repository can eliminate such inconsistencies and save storage space since each object would only appear once.

Furthermore, because providing support for each protocol may be cumbersome to developers of management  
30   applications, some of these developers have resorted to only supporting one or a few of these protocols. Thus, not all objects, i.e. network resources, may be

2025 RELEASE UNDER E.O. 14176

Docket No. AUS000153US1

represented in each repository. As a result queries for objects not handled by the a particular protocol will not be able to be completed.

- Therefore, it would be advantageous to have a
- 5 method and an apparatus that provides a common repository for all OID-based objects regardless of the protocol scheme and yet still allow queries from each of a variety of protocols, such as SNMP, LDAP, and CIM/XML.
- Furthermore, it would be advantageous to make the
- 10 repository easily expandable to accept existing OID-based data trees from a current repository and seamlessly integrate the new repository into the combined logical repository.

001650 1 148650

**SUMMARY OF THE INVENTION**

5 A method and apparatus is presented for maintaining a logical composite repository of Object Identifier (OID) tree structures on a server in a distributed data processing system. Each OID tree structure has been programmed to interface with an application programming interface (API) associated with an OID abstraction layer  
10 for the composite repository. An OID subtree structure can be added to the composite repository resulting in registration with the OID abstraction layer and, in addition, an OID subtree structure can be removed from the composite repository resulting in removal of the OID  
15 subtree structure from the registry associated with the OID abstraction layer.

Any query from a requester in the distributed data processing system about an object contained in the logical composite repository associated with the server  
20 is processed by the OID abstraction layer. The query must be in a protocol, such as SNMP, LDAP, and CIM/XML, recognized by the OID abstraction layer. The repository associated with the object of the query is determined from the OID abstraction layer registry. The query is  
25 formatted to be consistent with the API associated with the OID abstraction layer and sent to the repository associated with the object. When a reply is received from the repository, it is formatted in the protocol of the original query and sent to the requester in the  
30 distributed data processing system.

**BRIEF DESCRIPTION OF THE DRAWINGS**

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

**Figure 1** is an exemplary diagram of a distributed data processing system in which the present invention may be implemented;

**Figure 2** is an exemplary block diagram of a server in which the present invention may be implemented;

**Figure 3A** is an exemplary diagram of a standard object identifier tree illustrating the naming of nodes;

**Figure 3B** is an exemplary diagram of a OID tree structure in accordance with the present invention;

**Figure 4** is an exemplary block diagram showing the relationship between the OID Abstraction Layer to external queries and a logical composite repository; and

**Figure 5** is a flowchart outlining an exemplary operation of the present invention.

2025 RELEASE UNDER E.O. 14176

**DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT**

With reference now to the figures, **Figure 1** depicts a  
5 pictorial representation of a distributed data processing  
system. Distributed data processing system **100** is a  
network of computers in which the present invention may be  
implemented. Distributed data processing system **100**  
contains a network **102**, which is the medium used to  
10 provide communications links between various devices and  
computers connected together within distributed data  
processing system **100**. Network **102** may include permanent  
connections, such as wire or fiber optic cables, or  
temporary connections made through telephone connections.

15 In the depicted example, a server **104** is connected to  
network **102** along with storage unit **106**. In addition,  
clients **108**, **110**, and **112** also are connected to network  
**102**. These clients **108**, **110**, and **112** may be, for example,  
personal computers or network computers. For purposes of  
20 this application, a network computer is any computer,  
coupled to a network, which receives a program or other  
application from another computer coupled to the network.  
In the depicted example, server **104** provides data, such as  
boot files, operating system images, and applications to  
25 clients **108-112**. Clients **108**, **110**, and **112** are clients to  
server **104**. Distributed data processing system **100** may  
include additional servers, clients, and other devices not  
shown.

In the depicted example, distributed data processing  
30 system **100** is the Internet with network **102** representing a  
worldwide collection of networks and gateways that use the

TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, distributed data processing system **100** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN).

While a management "server" is shown in **Figure 1**, the management functions of the present invention may be performed by a management application resident on any type computing device connected to the network **102**.

These network devices are remotely managed using either a standard protocol, such as SNMP, LDAP, and CIM/XML, or another equivalent management protocol. Queries can be submitted using the supported protocols to the server machines.

**Figure 1** is intended as an example, and not as an architectural limitation for the present invention. As may be readily apparent to those of ordinary skill in the art, many other types of devices may be connected to the network **102** without departing from the spirit and scope of the present invention. For example, the network **102** may provide a communication pathway for client devices to send and receive data from printers, plotters, scanners, multiple drive libraries, and the like.

The OID abstraction layer identifies the object in the query, determines a repository in which the object is represented, and sends a request to the repository for the  
15 desired object via the API. The repository responds to the API with the object information which is placed into a reply message formatted for the proper protocol and sent to the requesting application.

Referring to **Figure 2**, a block diagram of a data processing system that may be implemented as a management server or a managed server, such as management server **114** or managed server **104** in **Figure 1**, is depicted in accordance with a preferred embodiment of the present invention. Data processing system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory

Docket No. AUS000153US1

controller/cache **208** and I/O bus bridge **210** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. A number of modems may be connected to PCI bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers **108-112** in **Figure 1** may be provided through modem **218** and network adapter **220** connected to PCI local bus **216** through add-in boards.

Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, data processing system **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 2** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention. The data processing system depicted in **Figure 2** may be, for example, an IBM RISC/System 6000 system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system.

Docket No. AUS000153US1

The management server, such as management server **114**, includes one or more network device management applications used to remotely manage a plurality of network devices **104-112** over a network **102**. These one or more network device management applications may be stored, for example, in local memory **209**, for example, and used to control the operations of the processor **202** or **204** on a remotely managed server **104**. As mentioned above, in a preferred embodiment of the present invention, the management server **114** manages the network devices **104-112** using a protocol such as SNMP, LDAP, CIM/XML, or a proprietary scheme.

Since it is often necessary to support multiple protocols on the same server, in known systems, a separate repository and processing system must be maintained for each protocol. Furthermore, the same object might be referenced by different schemes and information may be changed in an inconsistent manner so that the same query made using different protocols produces different results.

The present invention provides an OID abstraction layer and API executed by the processor **202** and/or **204** of the managed server **200**, for interpreting queries in various protocol formats, determining a repository in which requested object information is resident, sending a request message to the identified repository, and inserting the received object information into a message formatted for the appropriate protocol.

**Figure 3A** is an exemplary diagram of a standard object identifier (OID) tree data structure that can be processed by the present invention. **Figure 3A** is only intended to be a simplified depiction of a fictitious tree data structure that will aid in the understanding of

Docket No. AUS000153US1

the present invention and does not necessarily correspond to an actual standardized tree data structure. It should be noted that, while **Figure 3A** depicts some nodes as having a single branch, these nodes may contain additional branches which are not shown for clarity.

**Figure 3A** is only an example to illustrate the OID numbering scheme and a different tree data structures of more or less complicated architectures are possible.

As shown in **Figure 3A**, the nodes of the tree data structure designate "objects" and are represented by object identifiers (OIDs). An "object" in the context of the tree refers to an entry in the tree. These OIDs are often referred to by their human readable branch names rather than their numerical value. Thus, for example, an object OID of a tree may have a numerical value but be referred to as "Age" or "Name" or the like. The objects of the tree represent tables and record entries within tables. Thus, for example, a table "Person" may be comprised of records having entries corresponding to an index, a name, and an age.

An OID tree structure starts with a single root labeled 0. This is a unique label in the tree so it is always easy to identify the root of the tree. A node may have no children or it may have any number of children. If there are  $n$  child nodes, then these child nodes are labeled with integer values 1 through  $n$ . Following the path from the root to a particular node gives a unique numeric name for the node. For example, the node **302** can be reached via root 0, the root's child 1, 1's child 2, and 2's child 1. This path is indicated by starting at the root and sequencing each node number, separated by a

2025 RELEASE UNDER E.O. 14176

dot, until the desired node is reached. So node **302** is identified as "0.1.2.1".

5

15

30



Docket No. AUS000153US1

stored in secondary storage. It is well suited for information that is more static in nature, such as IP addresses associated with particular network interface cards. The advantage of LDAP is that complex, relational queries are possible. The present invention allows multiple repositories to coexist in a single logical composite repository where potential inconsistencies are eliminated.

CIM is a very general data model used to represent any type of information and, in itself, is independent of any protocol. XML (eXtensible Markup Language) is often used as a transport for CIM data, so, as a protocol, one refers to CIM/XML.

The present invention provides a mechanism by which each of these protocols is able to be used with objects in various repositories regardless of the particular format the repository supports. These repositories may be separate repositories or a combined repository.

With reference now to **Figure 4**, an exemplary functional block diagram illustrating a managed server is provided. As shown in **Figure 4**, the managed server includes an OID abstraction layer **414**, a corresponding API **416**, various repositories **408-412** and protocol interfaces **402-406**. The elements shown in **Figure 4**, in a preferred embodiment, are implemented as software executing on a managed server, such as server **200**. Of course, some or all of these elements may also embodied as hardware elements without departing from the spirit and scope of the present invention as will be readily apparent to those of ordinary skill in the art.

As shown in **Figure 4**, three protocol interfaces are provided through which queries may be received and reply

## Abstraction Layer through the API 416.

The repositories **408-412** include data structures that store the object information as well as a software application that may be used to search and retrieve information from the data structures. These software applications are provided with the ability to communicate with the API **416**. Each repository must be programmed to work with this API, regardless of the protocol or protocols supported by the repository. This protocol support is established when the subtree is registered with the OID abstraction layer **414**. OID Abstraction Layer **414** maps all incoming requests, regardless of protocol, into an API request that the attached repository can understand.

When a request is received from API **416**, the repository understands the request, searches the data structure for the requested object information, and sends a reply message to the API **416** with the requested object information. The API **416** provides the requested object information to the OID Abstraction Layer **414**. The OID Abstraction Layer **414** then generates a reply message to the application that sent the request. The reply message is formatted for the particular protocol used in the request message from the requesting application. The reply message is then sent to the appropriate protocol interface **402-406**.

As a more specific example, suppose that an SNMP query arrives via the SNMP interface **404** and the query deals with information in Repository #2 **410** which has been registered to support the LDAP protocol. SNMP is a very simple protocol with operations like get, getnext, set, and so forth. The SNMP query is mapped into the API query and sent to the repository #2. When the response to

Docket No. AUS000153US1

the query is returned to OID Abstraction Layer **414**, it is formatted into an SNMP response and sent back to the source of the query.

Not every protocol operation can be mapped into an equivalent operation in another protocol. For example, LDAP and CIM/XML can store information about relationships between nodes. Suppose Repository #2 **410** contains information about five users and ten printers, and has registered with the OID abstraction layer as supporting the LDAP protocol. An LDAP query of "What printers are associated with user X" can be answered. However, SNMP does not contain this type of relational structure, so the objects in Repository #1 **408**, which have been registered as being able to support only the SNMP protocol, would not be able to respond to this type of query. If the OID Abstraction **414** received an LDAP query of the form "What printers are associated with User Y" where User Y is contained in Repository #1 **408**, OID Abstraction Layer **414** would have to respond that the query cannot be satisfied.

When a repository is added to the set of repositories, it is registered with OID Abstraction Layer **414** based on the protocol capabilities it can support. The registry provides information identifying the anchor points in the OID tree structure maintained by the repository. These anchor points are the objects in the OID tree structure that define a subtree of objects and their attributes. If a query is received for an object that has an OID that is below this anchor point in the OID tree structure, the OID Abstraction Layer **414** is able to determine that the object information for the requested object must be in the repository that maintains

the object anchor point. In this way, the OID Abstraction Layer **414** is able to identify which repository maintains object information for a requested object.

15           A composite repository is beneficial in that only  
one repository has to maintain potentially multiple  
instances of single server resource. Thus, the problems  
associated with having multiple objects, supported by  
multiple repositories, referencing the same server  
20 resource are avoided. In addition, if a single  
repository is utilized, the need for the OID Abstraction  
Layer **414** to determine which repository maintains the  
object information would be greatly simplified.

Individual repositories can also be removed from the composite repository and the corresponding entries removed from the registry of OID Abstraction Layer **414**. For example, many devices include uninstall programs for use when the device is removed from a particular domain. If the device had previously been added to the registry of OID Abstraction Layer **414**, it is possible to have the uninstall program remove that entry.

Docket No. AUS000153US1

**Figure 5** is a flowchart outlining an exemplary operation of the present invention. As shown in **Figure 5**, the operation starts with the receipt of a request message from an application in a particular communication protocol (step **510**). The object information in the request is identified (step **520**) and a repository that maintains this object information is identified based on a repository registry (step **530**). The incoming request must be mapped to an equivalent request using the API (step **535**). Once the repository is identified and the API request formulated, the request is sent to the repository for the object information (step **540**). A reply message is received from the repository with the requested object information (step **550**). The requested object information is then inserted into a reply message formatted to the protocol used by the requesting application and sent to the requesting application (step **560**). The operation then ends.

Thus, the present invention provides a mechanism by which object information for objects in various repositories may be accessed regardless of the particular protocol used. Furthermore, the present invention provides a mechanism by which a combined repository may be utilized by a plurality of applications using different communication protocols.

The description of the present invention has been presented for purposes of illustration and description but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention and

the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated

Docket No. AUS000153US1

**CLAIMS:**

What is claimed:

1. A method on a server in a distributed data  
5 processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the method comprising the steps of:
  - receiving, in an OID abstraction layer, an OID tree structure from a repository;
  - 10 registering the OID tree structure with a registry associated with the OID abstraction layer; and
  - adding the OID tree structure to a repository associated with the OID abstraction layer.
- 15 2. The method of claim 1, wherein the registry associated with the OID abstraction layer provides information identifying an anchor point in the OID subtree structure to be maintained by the repository.
- 20 3. The method of claim 2, wherein if the anchor point of the OID subtree structure is already registered with the OID abstraction layer, the registry is overwritten.
4. The method of claim 2, wherein if a query is  
25 received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the OID abstraction layer identifies a repository that maintains object information for the requested object based on the registered anchor point.
- 30 5. The method of claim 1, wherein the repository is configured such that the repository recognizes requests

from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

6. The method of claim 5, wherein the OID abstraction layer receives the information retrieved from the repository through the API and encapsulates the information in a reply message to a protocol interface.

7. The method of claim 1, wherein the OID abstraction layer receives a request for object data from a protocol interface, converts the request into an application program interface (API) request which is forwarded to the repository, and receives an API reply from the repository having the object data.

8. The method of claim 7, wherein the OID abstraction layer reformats the object data in a reply message and sends the reply message to the protocol interface.

9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:

- receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer;

locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

Docket No. AUS000153US1

retrieving the object data from the repository using an OID abstraction layer application program interface (API).

- 5 10. The method of claim 9, wherein the first query is mapped into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.
- 10 11. The method of claim 10, wherein if the first query cannot be mapped into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.
- 15 12. The method of claim 10, wherein the second query is sent to the repository that contains the object associated with the first query.
- 20 13. The method of claim 12, wherein a first reply is received at the API associated with the OID abstraction layer from the repository that contains the object associated with the first query.
- 25 14. The method of claim 13, wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.
- 30 15. The method of claim 14, wherein the second reply is sent to the requester in the distributed data processing system.

Docket No. AUS000153US1

16. The method of claim 9, wherein each repository in the plurality of repositories contains information representing an Object Identifier (OID) subtree  
5 structure.
17. The method of claim 9, wherein Simple Network Management Protocol (SNMP) is a protocol recognized by the OID abstraction layer.
- 10 18. The method of claim 9, wherein Lightweight Directory Access Protocol (LDAP) is a protocol recognized by the OID abstraction layer.
- 15 19. The method of claim 9, wherein Common Information Model used in conjunction with eXtensible Markup Language (CIM/XML) is a protocol recognized by the OID abstraction layer.
- 20 20. An apparatus on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the apparatus comprising:
- an OID abstraction layer that receives an OID tree  
25 structure from a repository;
  - a registry, associated with the OID abstraction layer, that registers the OID tree structure; and
  - an adding means for adding the OID tree structure to a repository associated with the OID abstraction layer.

30

Docket No. AUS000153US1

21. The apparatus of claim 20, wherein the registry provides information identifying an anchor point in the OID tree structure to be maintained by the repository.

5 22. The apparatus of claim 21, wherein if the anchor point of the OID tree structure is already registered in the registry, then the registry overwrites the previous entry.

10 23. The apparatus of claim 21, wherein, if the OID abstraction layer receives a query for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the registry in the OID abstraction layer identifies a repository that maintains  
15 object information for the requested object based on the registered anchor point.

20 24. The apparatus of claim 20, wherein the repository is configured such that the repository recognizes requests received from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

25 25. The apparatus of claim 24, wherein the OID abstraction layer receives the information retrieved from the repositories through the API and encapsulates the information in a reply message to a protocol interface.

30 26. The apparatus of claim 20, wherein the OID abstraction layer receives a request for object data from a protocol interface, converts the request into an

OFFICIAL FILE

Docket No. AUS000153US1

application program interface (API) request which is forwarded to the repository, and receives an API reply from the repository having the object data.

5 27. The apparatus of claim 26, wherein the OID abstraction layer encapsulates the object data in a reply message and sends the reply message to the protocol interface.

10 28. An apparatus on a server in a distributed data processing system for retrieving object data from a repository, comprising:

15 a receiving means for receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer;

20 a locating means for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

a retrieving means for retrieving the object data from the repository using an OID abstraction layer application program interface (API).

25 29. The apparatus of claim 28, further comprising a mapping means for mapping the first query into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

30 30. The apparatus of claim 29, wherein if the mapping means cannot map the first query into a second query due

Docket No. AUS000153US1

to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

- 5 31. The apparatus of claim 29, further comprising a first sending means, in the OID abstraction layer, that sends the second query to a repository that contains the object associated with the first query.
- 10 32. The apparatus of claim 31, wherein the retrieving means receives a first reply at the API from the repository that contains the object associated with the first query.
- 15 33. The apparatus of claim 32, further comprising a transforming means, in the OID abstraction layer, that transforms the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.
- 20 34. The apparatus of claim 33, further comprising a second sending means, in the OID abstraction layer, that sends the second reply to the requester in the distributed data processing system.
- 25 35. The apparatus of claim 28, wherein each repository contains Object Identifier (OID) tree structures.
- 30 36. The apparatus of claim 28, wherein the receiving means recognizes a Simple Network Management Protocol (SNMP) query.

37. The apparatus of claim 28, wherein the receiving means recognizes a Lightweight Directory Access Protocol (LDAP) query.

38. The apparatus of claim 28, wherein the receiving means recognizes a Common Information Model used in conjunction with eXtensible Markup Language (CIM/XML) query.

10 39. A computer program product in a computer readable  
medium for maintaining a repository of Object Identifier  
(OID) tree structures, comprising:

instructions for receiving, in an OID abstraction layer, an OID tree structure from a repository;

```

15      instructions for registering the OID tree structure
      with a registry associated with the OID abstraction
      layer; and

```

instructions for adding the OID tree structure to a repository associated with the OID abstraction layer.

20

40. The computer program product of claim 39, further comprising instructions for maintaining the registry associated with the OID abstraction layer and providing information identifying an anchor point in the OID tree

25 structure to be maintained by the repository.

41. The computer program product of claim 40, wherein if the anchor point of the OID tree structure is already registered with the OID abstraction layer, the instructions for registering overwrites the previous entry.

42. The computer program product of claim 40, further comprising instructions for identifying a repository that maintains object information for the requested object based on the registered anchor point if a query is received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure.

15

20

25

instructions for receiving an API reply from the subtree repository having the object data.

46. The computer program product of claim 45, further comprising instructions for encapsulating the object data

Docket No. AUS000153US1

in a reply message and sending the reply message to the protocol interface.

47. A computer program product in a computer readable  
5 medium for retrieving object data from a repository,  
comprising:

instructions for receiving a first query for the  
object data from a requester in the distributed data  
processing system, wherein the first query is in a  
10 protocol recognized by an OID abstraction layer;

instructions for locating a repository that contains  
the object data requested in the first query based on a  
registry associated with the OID abstraction layer; and

instructions for retrieving the object data from the  
15 repository using an OID abstraction layer application  
program interface (API).

48. The computer program product of claim 47, wherein  
the instructions for receiving the first query map the  
20 first query into a second query, wherein the second query  
is consistent with an application program interface (API)  
associated with the OID abstraction layer.

49. The computer program product of claim 48, wherein if  
25 the instructions for receiving the first query map cannot  
map the first query into a second query due to a  
limitation of the repository that contains the object  
associated with the first query, then the first query  
cannot be satisfied.

30

50. The computer program product of claim 48, further  
comprising instructions for sending the second query to

the repository that contains the object associated with the first query.

51. The computer program product of claim 50, further  
5 comprising instructions for receiving a first reply at  
the API associated with the OID abstraction layer from  
the repository that contains the object associated with  
the first query.

52. The computer program product of claim 51, further comprising instructions for transforming the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.

15

53. The computer program product of claim 52, further comprising instructions for sending the second reply to the requester in the distributed data processing system.

20 54. The computer program product of claim 47, wherein  
the repository contains Object Identifier (OID) tree  
structures.

55. The computer program product of claim 47, wherein  
25 instructions for receiving a first query recognize a  
Simple Network Management Protocol (SNMP) query.

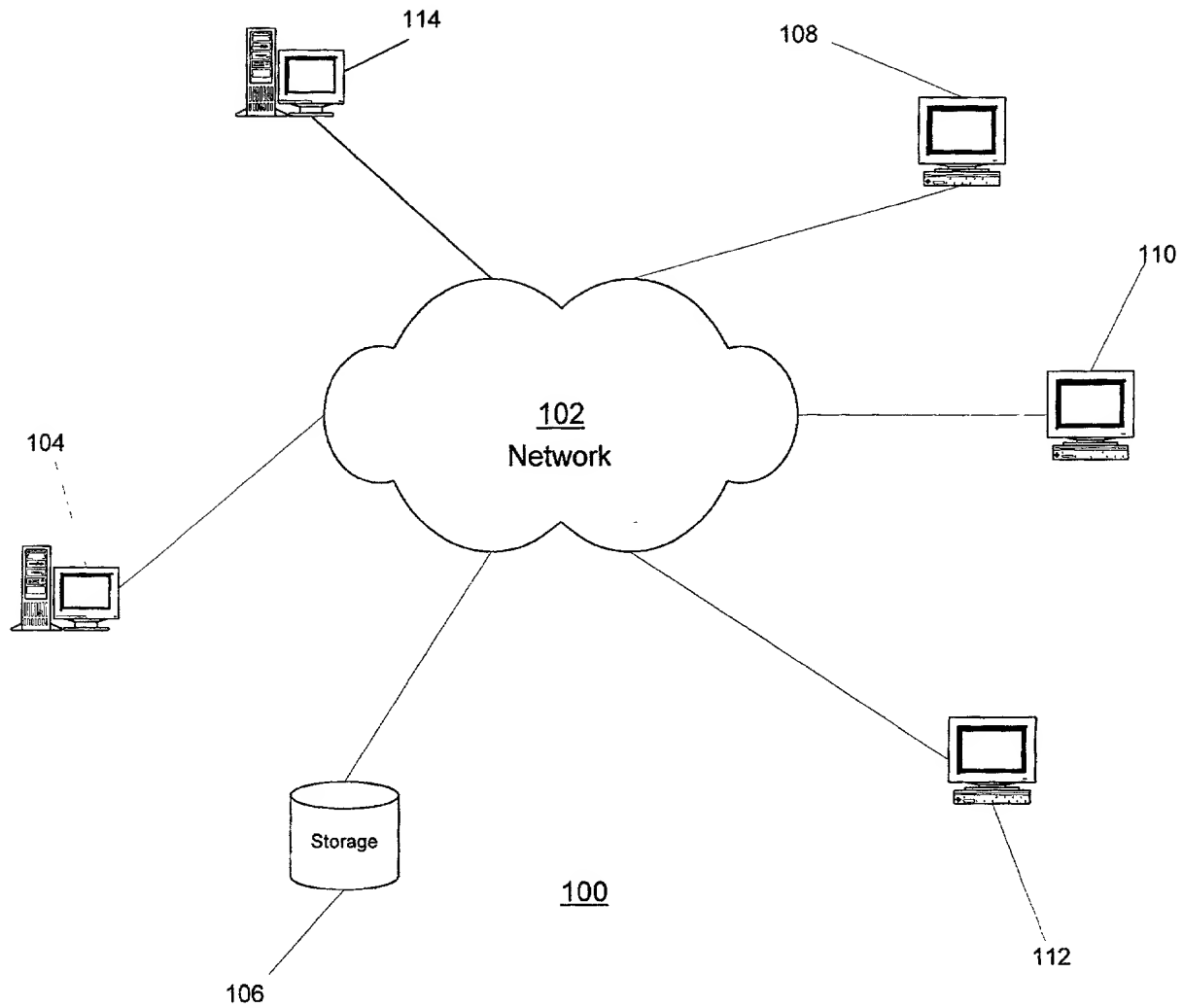
56. The computer program product of claim 47, wherein  
instructions for receiving a first query recognize a  
30 Lightweight Directory Access Protocol (LDAP) query.

57. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Common Information Model used in conjunction with eXtensible Markup Language (CIM/XML) query.

5

**ABSTRACT OF THE DISCLOSURE****A METHOD AND APPARATUS FOR BRIDGING SERVICE FOR STANDARD  
5 OBJECT IDENTIFIER BASED SYSTEMS**

A method and apparatus is presented for maintaining a logical composite repository of Object Identifier (OID) tree structures on a server in a distributed data  
10 processing system. Each OID subtree repository has been programmed to interface with an application programming interface (API) associated with an OID abstraction layer for the logical composite repository. An OID subtree structure can be added to the logical composite  
15 repository resulting in registration with the OID abstraction layer. Any query from a requester in the distributed data processing system about an object contained in the logical composite repository associated with the server is processed by the OID abstraction  
20 layer. The query must be in a protocol, such as SNMP, LDAP, and CIM/XML, recognized by the OID abstraction layer. The repository associated with the object of the query is determined from the OID abstraction layer registry. The query is formatted to be consistent with  
25 the API associated with the OID abstraction layer and sent to the repository associated with the object. When a reply is received from the repository, it is formatted in the protocol of the original query and sent to the requester in the distributed data processing system.



100  
Network  
**Figure 1**

AUS000153US1

200  
Figure 2

AUS000153US1



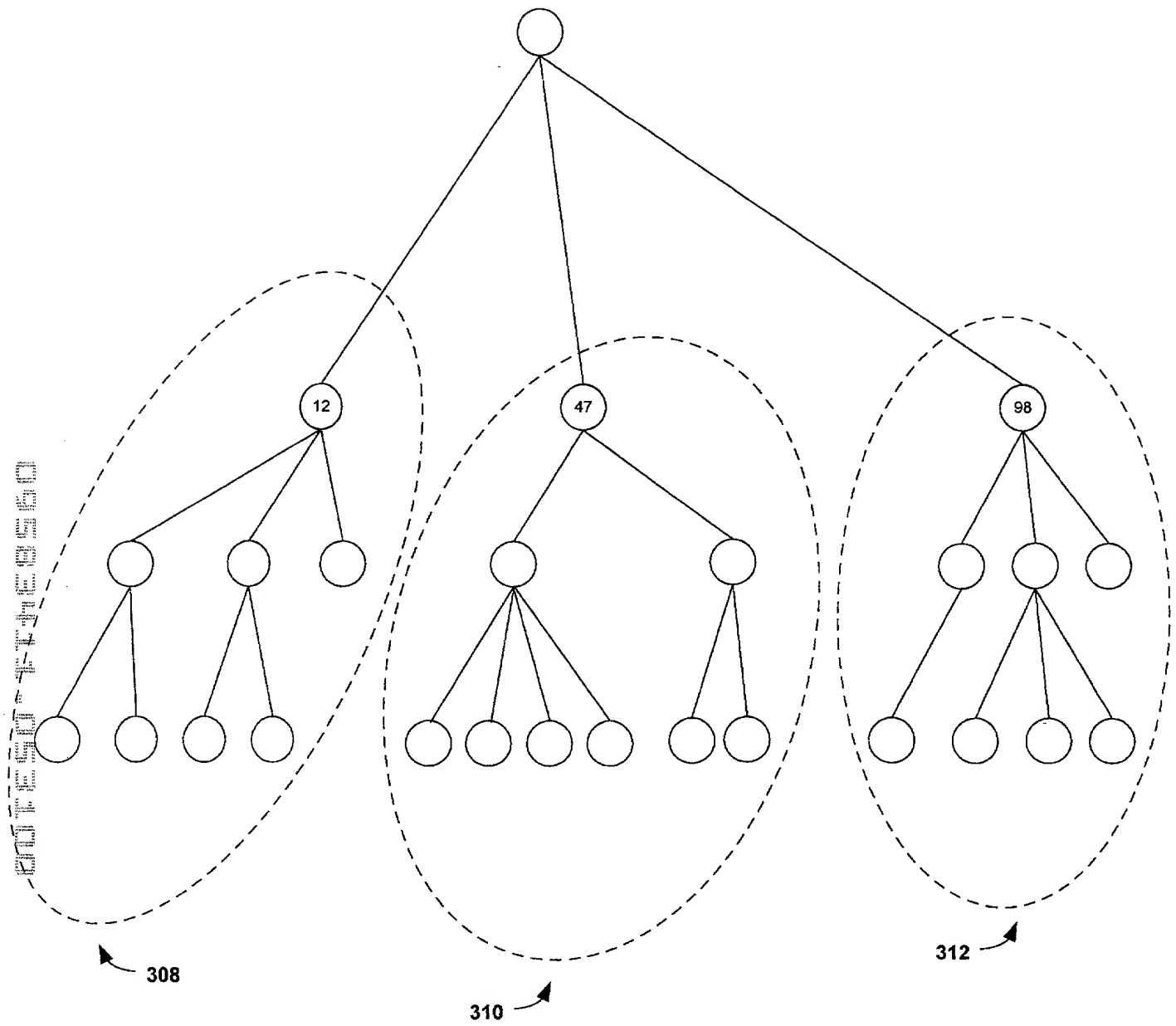
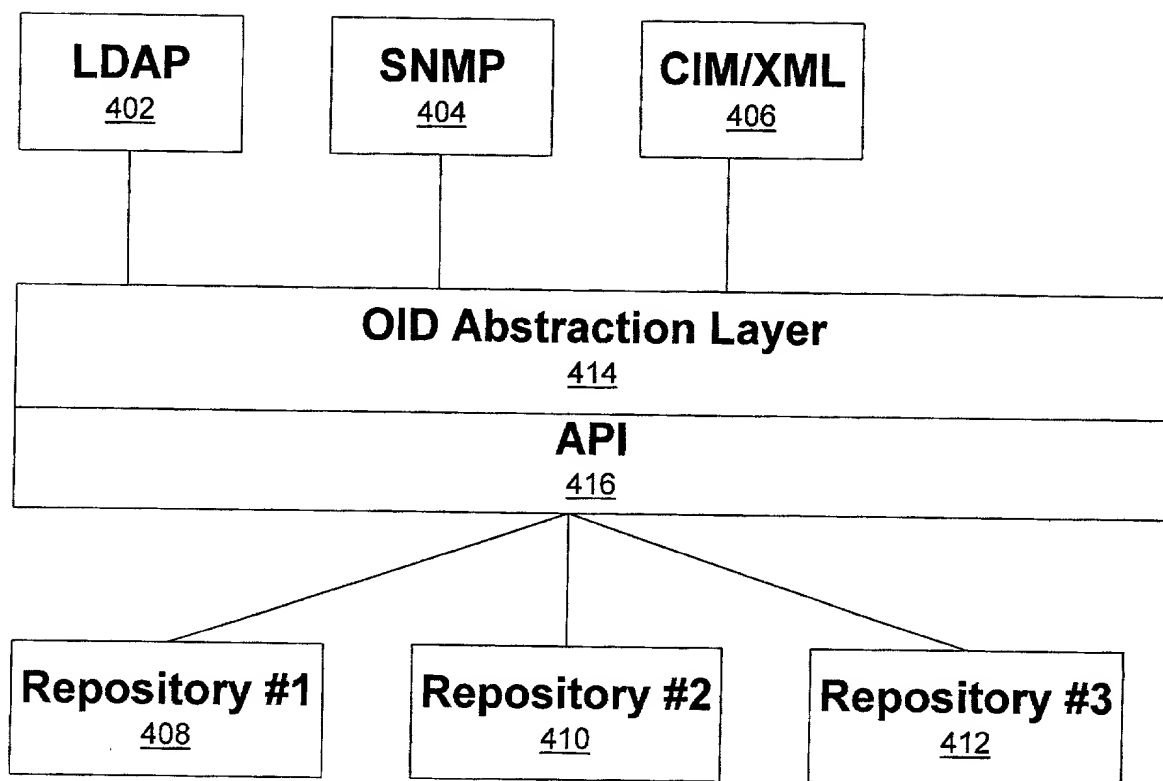


Figure 3B

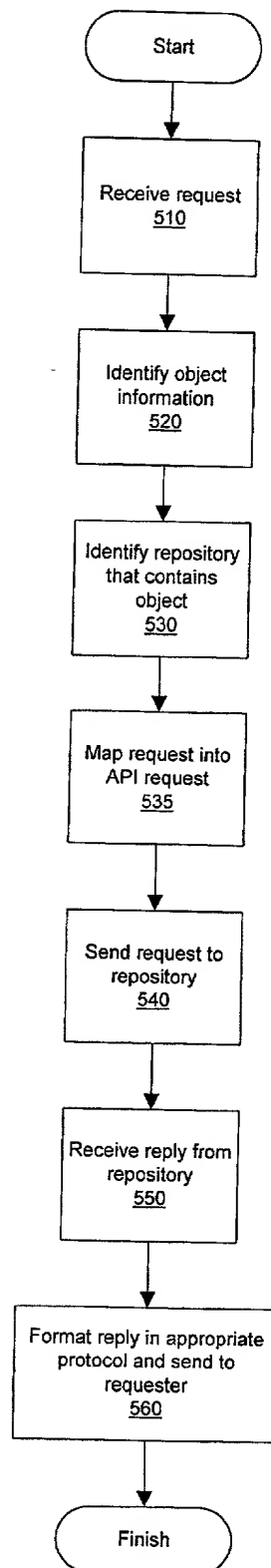
AUS000153US1



**Figure 4**

AUS000153US1

AUS000153US1



**DECLARATION AND POWER OF ATTORNEY FOR  
PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

**A METHOD AND APPARATUS FOR BRIDGING SERVICE FOR STANDARD OBJECT IDENTIFIER  
BASED PROTOCOLS**

the specification of which (check one)

X is attached hereto.

\_\_\_ was filed on \_\_\_\_\_  
as Application Serial No. \_\_\_\_\_  
and was amended on \_\_\_\_\_  
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):	Priority Claimed
_____ (Number)	____ Yes ____ No
_____ (Country)	
_____ (Day/Month/Year)	

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial #)	(Filing Date)	(Status)
------------------------	---------------	----------

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; Thomas E. Tyson, Reg. No. 28,543; James H. Barksdale, Jr., Reg. No. 24,091; Casimer K. Salys, Reg. No. 28,900; Robert M. Carwell, Reg. No. 28,499; Douglas H. Lefevre, Reg. No. 26,193; Jeffrey S. LaBaw, Reg. No. 31,633; David A. Mims, Jr., Reg. No. 32,708; Volel Emile, Reg. No. 39,969; Anthony V. England, Reg. No. 35,129; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; Duke W. Yee, Reg. No. 34,285; Colin P. Cahoon, Reg. No. 38,836; Rudolph J. Buchel, Reg. No. 43,448; Stephen R. Loe, Reg. No. 43,757; Stephen J. Walder, Jr., Reg. No. 41,534; Charles D. Stepps, Jr., Reg. No. 45,880; and Stephen R. Tkacs, Reg. No. P-46,430.

Send correspondence to: Duke W. Yee, Carstens, Yee & Cahoon, LLP, P.O. Box 802334, Dallas, Texas 75380 and direct all telephone calls to Duke W. Yee, (972) 367-2001

FULL NAME OF SOLE OR FIRST INVENTOR: Kurt Russell Taylor

INVENTORS SIGNATURE: [Signature] DATE: 5/30/2000

RESIDENCE: 9010 Brimstone Lane  
Austin, Texas 78747

CITIZENSHIP: United States

POST OFFICE ADDRESS: SAME AS ABOVE